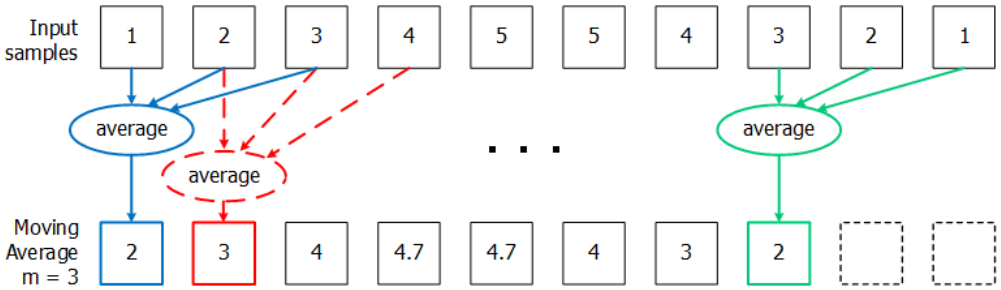
Session d'exercices – Pointeurs
Moyenne glissante

En statistiques, une **moyenne glissante** (*moving average* en anglais), consiste en la création d'une série de moyennes de différents sous-ensembles d'un ensemble de données. En *digital signal processing*, la moyenne glissante est souvent utilisée comme filtre sur un signal d'entrée.

Ci-dessous, nous avons deux exemples de calcul de moyenne glissante. Pour chacun des exemples, la séquence d'entrée contient les dix éléments suivants (**échantillons**) 1, 2, 3, 4, 5, 5, 4, 3, 2, 1, ainsi que la taille du sous-ensemble utilisé pour la moyenne glissante `m = 3`.

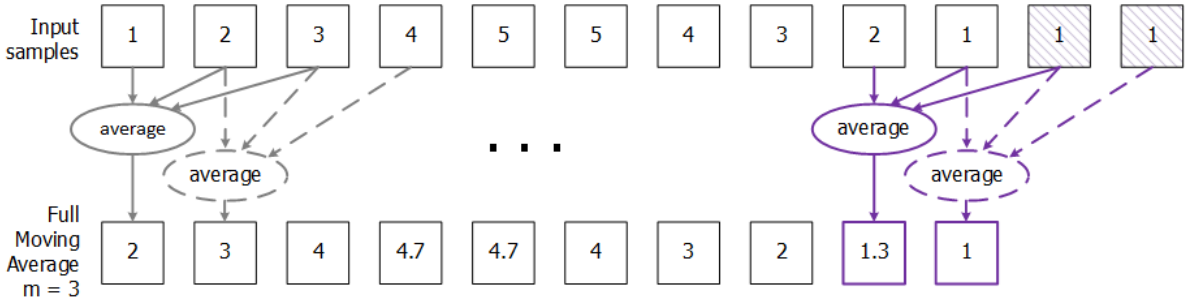
Example 1 : Simple moving average

La séquence de sortie (le résultat) est obtenue de la manière suivante : la moyenne du premier sous-ensemble est égale au premier échantillon de sortie ($average(1, 2, 3) = 2$). La moyenne du sous-ensemble suivant est la valeur du second échantillon de sortie ($average(2, 3, 4) = 3$), etc. La moyenne du dernier sous-ensemble est la valeur du dernier échantillon. Cela a pour conséquence de produire une séquence de sortie qui contient deux éléments de moins que la séquence d'entrée.



Example 2 : Full moving average

Ici, la séquence de sortie a la même taille que la séquence d'entrée. Supposez que la dernière valeur d'entrée est répétée autant de fois que nécessaire. Les valeurs des deux derniers échantillons de sortie (les deux manquants dans l'exemple précédent) sont alors $average(2, 1, 1) = 1.3$ et $average(1, 1, 1) = 1$, respectivement.



Dans cet exercice, vous devez écrire un programme qui, pour une séquence d'entrées donnée, crée deux séquences de sortie : la première séquence est calculée avec la méthode du **simple moving average**, la deuxième avec celle du **full moving average**. Suivez les instructions dans la page suivante.

a) [Difficulté : **] Écrivez la fonction

`void movAvg(float *in, float *out, int m)`,

qui prend comme arguments les adresses de début du tableau de données et du tableau contenant les moyennes, ainsi que la taille du groupe d'éléments utilisés pour calculer chaque moyenne. Cette fonction remplit le tableau des moyennes suivant le premier exemple (simple moving average) et affiche tous les éléments de ce tableau comme dans l'exemple de sortie au bas de cette page.

b) [Difficulté : ***] Écrivez la fonction

`void movAvgFull(float *in, float *out, int m)`,

qui prend comme arguments les adresses de début du tableau de données et du tableau contenant les moyennes, ainsi que la taille du groupe d'éléments utilisés pour calculer chaque moyenne. Cette fonction remplit le tableau des moyennes suivant le deuxième exemple (full moving average) et affiche tous les éléments de ce tableau comme dans l'exemple de sortie au bas de cette page.

Continuez en écrivant tout ce qui manque pour compléter le programme, incluant ce qui suit :

c) [Difficulté : *] Déclarez et initialisez un tableau de données de 10 éléments en virgule flottante dont les valeurs sont : 1, 2, 3, 4, 5, 5, 4, 3, 2 et 1.

d) [Difficulté : *] Votre programme doit être appelé avec un paramètre. Ce paramètre est le nombre d'éléments utilisés pour calculer les moyennes. (hint : il faut utiliser `argc/argv`, et pas `scanf` !)
   — Si ce paramètre n'est pas spécifié ou plusieurs paramètres sont spécifiés, le programme doit afficher le message

   Incorrect number of program arguments.

   et ensuite terminer.
   — Si la valeur de ce paramètre est plus petite que 1 ou plus grande que la taille du tableau de données, le programme doit afficher le message

   Invalid program argument value.

   et ensuite terminer.

e) [Difficulté : *] Le programme doit faire appel aux fonctions `movAvg` et `movAvgFull`.

Exemple de sortie du programme pour `m = 3` :

```
stojilov@IC-CO-IN-SC291:~/myfiles/[redacted]$ ./MovAvg 3


Moving average for m = 3...
2.0 3.0 4.0 4.7 4.7 4.0 3.0 2.0

Full moving average for m = 3...
2.0 3.0 4.0 4.7 4.7 4.0 3.0 2.0 1.3 1.0
```